VŠB - TECHNICKÁ UNIVERZITA OSTRAVA
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

BAKALÁŘSKÁ PRÁCE

2013 **Ladislav Folta**

VŠB - TECHNICKÁ UNIVERZITA OSTRAVA
FAKULTA ELEKTROTECHNIKY A INFORMATIKY
KATEDRA INFORMATIKY

Absolvování individuální odborné praxe

Individual professional practice in the company

2013 **Ladislav Folta**

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Ladislav Folta**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Absolvování individuální odborné praxe
Individual Professional Practice in the Company

Zásady pro vypracování:

1. Student vykoná individuální praxi ve firmě: Profiq s.r.o.
2. Struktura závěrečné zprávy:
a) Popis odborného zaměření firmy, u které student vykonal odbornou praxi a popis pracovního zařazení studenta
b) Seznam úkolů zadaných studentovi v průběhu odborné praxe s vyjádřením jejich časové náročnosti
c) Zvolený postup řešení zadaných úkolů
d) Teoretické a praktické znalosti a dovednosti získané v průběhu studia uplatněné studentem v průběhu odborné praxe
e) Znalosti či dovednosti scházející studentovi v průběhu odborné praxe
f) Dosažené výsledky v průběhu odborné praxe a její celkové zhodnocení

Seznam doporučené odborné literatury:

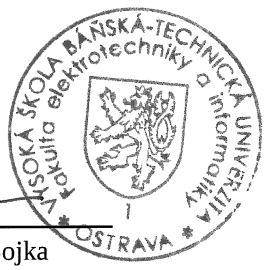
Podle pokynů konzultanta, který vedl odbornou praxi studenta.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Zdeněk Sawa, Ph.D.**

Konzultant bakalářské práce: Ing. Rastislav Kanócz

Datum zadání: 16.11.2012
Datum odevzdání: 07.05.2013



doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě, dne 6.5.2013

Podpis: Ladislav Fok

Tímto bych rád poděkoval mému konzultantu panu Ing. Rastislavu Kanóczovi za výtečné vedení při vykonávání bakalářské praxe, vedoucímu této práce panu Ing. Zdeňku Sawovi, Ph.D. za umožnění absolvování této praxe a v neposlední řadě také všem mým spolupracovníkům a rodině za jejich podporu.

Abstrakt

Tento dokument je zprávou o absolvování individuální bakalářské praxe ve firmě a popisuje práci na dvou Open Source projektech. Popisuje zadané úkoly a vysvětluje jejich řešení. Dokument také obsahuje teoretické a praktické znalosti a dovednosti získané v průběhu studia a uplatněné při práci na těchto projektech a porovnává je se znalostmi a zkušenostmi získanými v průběhu absolvování praxe. Dále se tento dokument bude zabývat dosaženými výsledky a jejich hodnocením.

Klíčová slova: Individuální odborná praxe, bakalářská práce, Open Source, Open Identity Manager, Identity Management, Identity Connector Framework, OpenIDM, OpenICF, Reverzní inženýrství, Robot Framework, Testování software, Automatizace testů

Abstract

This document provides results of absolving individual practice in a company and describes work on two Open Source projects. Document describes given tasks and explains their solution. Furthermore it also contains comparison between theoretical and practical knowledge obtained by studying and theoretical and practical knowledge earned during work in the company. In addition, this document will address the results and their evaluation.

Keywords: Individual professional practice in the company, Bachelor thesis, Open Source, Open Identity Manager, Identity Management, Identity Connector Framework, OpenIDM, OpenICF, Reverse Engineering, Robot Framework, Software testing, Test automation

Seznam použitých zkratek

OpenIDM	Open Identity Manager
OpenICF	Open Identity Connector Framework
LDAP	Light-weight Directory Access Protocol
CSV	Comma Separated Value
XML	Extensible Markup Language
ISTQB	International Software Testing Qualifications Board
TMMi	Testing Maturity Model
CentOS	Community Enterprise Operating System
RHEL	Red Hat Enterprise Linux
JSON	JavaScript Object Notation

Obsah

1 Úvod

Cílem této práce je seznámit čtenáře s průběhem odborné praxe, kterou jsem vykonával ve společnosti profiq s.r.o. V následujících kapitolách bude popsáno, čím se firma zabývá, na jakých projektech jsem pracoval a také zde budou popsány jednotlivé mně zadané úkoly a jejich řešení. V této práci se také budu zabývat tím, jaké teoretické a praktické znalosti nabyté v průběhu studia jsem uplatnil při vykonávání této praxe a naopak jaké dovednosti a znalosti mi při vykonávání praxe chyběly. Závěr této zprávy se bude zabývat dosaženými výsledky v průběhu praxe a také celkovým hodnocením.

2 Odborné zaměření firmy profiq s.r.o a pracovní zařazení studenta

Firma profiq s.r.o. [1] se zabývá poskytováním služeb v oblasti testování software a v oblasti integrace řešení identity managementu.

2.1 Služby v oblasti testování software

V oblasti testování software firma profiq s.r.o poskytuje následující služby:

- Odborné konzultace - firma poskytuje odborné konzultace zákazníkům v oblastech testovacích metodik, procesů testování, strategiích testování, v jakém prostředí a jakými nástroji je vhodné jejich produkt testovat a v neposlední řadě také v oblasti automatizace testování
- Školení v oblasti testování založené na standardech ISTQB a TMMi
- Poskytuje jednotlivé specialisty nebo celé týmy a řešení pro testování produktů zákazníkům

2.2 Služby v oblasti integrace řešení Identity Managementu

V této oblasti firma poskytuje řešení založené na Open Source technologiích, konkrétně Open Identity Stack společnosti ForgeRock AS. [2]

2.3 Mé pracovní zařazení

Během vykonávání praxe jsem ve firmě pracoval na pozici junior software inženýra. Konkrétně jsem pracoval na projektech Open Identity Manager (OpenIDM) [3] a Open Identity Connector Framework (OpenICF) [4] společnosti ForgeRock AS v oddělení testování software (QA oddělení).

3 Seznam úkolů zadaných studentovi v průběhu praxe

Jelikož jsem v průběhu praxe pracoval na dvou projektech, budou úkoly rozděleny do podkapitol ”Projekt OpenIDM” a ”Projekt OpenICF”

3.1 Identity a Access management

Aby čtenář měl lepší přehled o problematice, kterou projekty, na kterých jsem pracoval, řeší, zaměříme se nejprve na vysvětlení problematiky Identity a Access managementu.

Identity a Access management jsou velmi komplexní komponentou podnikových infrastruktur. Zaprvé proto, že jsou samy o sobě velmi složité. Zadruhé proto, že mají hodně závislostí na jiných systémech (databáze, portály, SOA, aplikační a web servery a další), a zatřetí také proto, že musí podporovat širokou škálu operačních systémů (Linux, Windows, Solaris, AIX, HP-UX, ...).

Identity a Access management (IAM) zahrnuje lidi (identity), procesy a produkty, které spravují jednotlivé identity a řídí přístup identit k firemním zdrojům. IAM řeší otázku: Kdo má přístup k jakým datům či aplikacím za jakých podmínek. IAM můžeme rozdělit do 4 hlavních komponent: autentizace (authentization), autorizace (authorization), správa uživatelů (user management) a centrální úložiště identit (central user repository). Komponenta autentizace vyžaduje, aby uživatel poskytl dostatečné údaje k tomu,

aby mu mohl být udělen přístup k systému nebo danému zdroji. Jakmile uživatel poskytne tyto údaje a je autentizován - neboli víme, kdo se přihlásil, vstupuje do hry další komponenta - autorizace. Autorizace má za úkol určit, zda daný uživatel má přístup k aplikaci, systému nebo datům, ke kterým se snaží přistoupit. Další komponentou je správa uživatelů, která se stará o administraci a správu životního cyklu jednotlivých identit. Poslední komponentou je komponenta centrálního úložiště identit. Tato komponenta ukládá a poskytuje informace o jednotlivých identitách a také poskytuje službu verifikace přístupových údajů. [11]

Typickými uživateli IAM jsou banky, telekomunikační společnosti, vládní organizace, středně velké a velké společnosti, které mají hodně zaměstnanců a také velké množství externích uživatelů přistupujících do jejich systémů. AIM je těmito společnostmi využíván, protože jim snižuje náklady na správu podnikových systémů a zároveň pomáhá zvyšovat jejich bezpečnost. Pokud si představíme, že jakýkoliv výpadek způsobuje škody v miliónech, je celkem zřejmé, že vysoká kvalita těchto produktů je jejich základním požadavkem.

3.2 Projekt OpenIDM

3.2.1 Popis produktu

Open Identity Manager, zkráceně OpenIDM, společnosti ForgeRock AS je Open Source Identity Management řešením. OpenIDM umožňuje správu uživatelských účtů/identit, přístupových práv uživatelů v aplikacích a také správu životního cyklu identit. OpenIDM je platformě nezávislé, funguje tedy na různých operačních systémech, nad cloudem, sociálními platformami, v enterprise a mobilních prostředích. [10]

3.2.2 Zadané úkoly

Protože jsem se dosud nikdy nesetkal s pojmem identity middle-ware, mým prvním úkolem bylo nastudovat si problematiku, kterou identity middle-ware řeší. Poté jsem měl za úkol seznámit se s konkrétním identity middle-ware řešením - tedy s produktem OpenIDM. V době, kdy jsem byl s OpenIDM již poměrně obeznámen, začal jsem pracovat na testování tohoto produktu. Z počátku manuálně, podle již definovaných testovacích případů, později pomocí automatizovaného frameworku, na jehož tvorbě jsem se podílel.

Ruční testování

Jedním z úkolů při manuálním testování bylo ověřit funkčnost ukázkových příkladů [5] pro připravované vydání verze OpenIDM 2.1, které bylo oficiálně vydáno na konci ledna 2013.

Automatizace

Dalším ze zadaných úkolů bylo otestovat funkčnost OpenIDM filtrů pro sestavování dotazů (AND, OR, GREATERTHAN a další), a to nejprve manuálně a poté celý proces automatizovat.

3.3 Projekt OpenICF

3.3.1 Popis produktu

Open Identity Connector Framework (OpenICF) je další z řady produktů společnosti ForgeRock AS. Jak již jeho název sám napovídá, je Open Identity Connector Framework, rovněž jako

OpenIDM, Open Source řešením. OpenICF poskytuje jednotný interface pro komunikaci s různými druhy datových úložišť. Od těch nejjednodušších, jako je XML nebo CSV soubor, přes různé LDAP a SQL databáze až po cloudová řešení.

3.3.2 Zadané úkoly

Mým prvním úkolem pro OpenICF bylo obeznámit se s jeho použitím v OpenIDM a jeho možnostmi. V rámci seznamování se s Identity Connector Frameworkem jsem se měl obeznámit také s konektory, oficiálně podporovanými ForgeRockem. Po tomto obeznámení se s OpenICF a konektory jsem začal pracovat na reverzním inženýrství (reverse engineering) Contract Test Frameworku, který ForgeRock převzal z dnes již neexistující společnosti Sun Microsystems.

Ruční testování

Seznamování popisované výše probíhalo manuálním testováním konektorů. Toto manuální testování probíhalo společně s ověřováním funkčnosti ukázkových příkladů zmíněných v části manuálního testování OpenIDM. Testovány byly konektory dodávané společně s OpenIDM, tedy konektory: LDAP, ScriptedSQL, XML a CSV.

Dalším z úkolů manuálního testování bylo otestovat konektor vyvíjený pro připojení k Microsoft Active Directory.

Automatizace

Práce na automatizaci spočívala v provedení reverzního inženýrství Contract Test Frameworku. Cílem bylo přijít na to, jak tento framework vlastně pracuje, zdokumentovat existující testy a také zdokumentovat, které oblasti tyto testy pokrývají a popřípadě navrhnout testy nové, které by rozšířily stávající pokrytí.

4 Řešení zadaných úkolů

4.1 OpenIDM a OpenICF první kroky

Problematiku Identity middle-ware jsem studoval z důvěrných materiálů, které mi byly za tímto účelem poskytnuty. Po prostudování těchto materiálů jsem začal pracovat s konkrétním produktem a to s produktem OpenIDM. K osvojení si produktu OpenIDM jsem využil instalačního [5] a integračního [6] průvodce společnosti ForgeRock. Během procesu osvojování jsem rovněž prováděl prvotní testování na nejnovějších sestaveních (tzv. Nightly Buildech) a vefirikaci stávající projektové dokumentace - jak instalačního průvodce, tak průvodce pro integrátory a mnohých jiných dokumentů. A protože OpenICF a jednotlivé konektory jsou nedílnou součástí OpenIDM, získal jsem během práce s OpenIDM i znalosti týkající se OpenICF a jednotlivých konektorů.

Nyní již jsem byl obeznámen s oběma produkty natolik, abych mohl začít provádět funkcionální testování, v jehož rámci jsem se zaměřil na ověření jednotlivých funkcí produktu OpenIDM a taky jeho schopnost spolupracovat s ostatními systémy tzv. interoperability testing, kde cílem byly hlavně systémy jako Open Identity Connector Framework (OpenICF), jednotlivé konektory a vzdálené systémy, ke kterým se OpenIDM připojuje pomocí OpenICF a konektorů. Testování spočívalo v manuální exekuci jednotlivých příkazů podle předem daného testovacího scénáře, zkoumání správné funkčnosti podle funkčních specifikací, identifikaci a řešení problémů tzv. troubleshooting, doporučení na jejich řešení a reportování problému vývojovému týmu. Díky tomuto testovaní se mé dosud relativně povrchní znalosti OpenIDM hodně prohloubily. Příklad testovacích scénářů naleznete jako ukázkové příklady v OpenIDM instalačním průvodci. [5]

4.2 Automatizace OpenIDM testů

Jako další přišla na řadu automatizace těchto testů, které dosud existovaly pouze v podobě, kdy tester musel provádět jednotlivé příkazy manuálně a na základě výsledků těchto příkazů byl schopen vyhodnotit, zda tento test byl úspěšný - tedy zda testovaná funkcionalita funguje správně, či nikoliv. Pro automatizaci jsme se rozhodli použít Robot Framework [8] v kombinaci s námi napsanými knihovnami v jazyce python [9]. Příklad jedné z napsaných knihoven můžete vidět na následujícím obrázku.

```
"""
Python Lib for installing, starting and stopping OpenIDM
"""
import subprocess
import shutil
import os
import zipfile
import time
import mysql
import requests
import signal

class startup_shutdown:

    def __init__(self, lib_directory, operating_system, debug):
        """ save the resources directory"""
        resources = os.path.join(lib_directory, '..', 'resources')
        self._resources = resources
        self._operating_system = operating_system
        self._debug = debug

    def deploy_openidm_python(self, openidm_dir, openidm_zip_path):
        """ clean previous install of openidm and deploy a new one """

        self.check_os_supported()
        startup_file = {'unix': 'startup.sh', 'windows': 'startup.bat'}
        startup_path = os.path.join(openidm_dir, startup_file[self._operating_system])

        # Extract the parent dir as openidm_dir end with openidm
        # and openidm folder will be created when we unzip
        (head, tail) = os.path.split(openidm_dir)
        # do it again if the opendj has a trailing backslash
        if tail == "":
            (head, tail) = os.path.split(head)

        # Unzip the file
        if not os.path.exists(head):
            os.makedirs(head)
        os.chdir(head)
        args = [ "unzip", openidm_zip_path]
        with open(os.devnull, "w") as invisible:
            subprocess.call(args, stdout=invisible)
        print "Successfully installed OpenIDM"

        # Empty felix_cache
        felix_cache = os.path.join(openidm_dir, 'felix-cache')
        if os.path.isdir(felix_cache):
            shutil.rmtree(felix_cache)

        # Remove the text-based OSGi console bundle
        # TODO the version number could change, we should look for a pattern
        felix_shell = os.path.join(openidm_dir, 'bundle', 'org.apache.felix.shell.tui-1.4.1.jar')
        if os.path.exists(felix_shell):
            os.remove(felix_shell)
```

Obrázek 1: Ukázka kódu knihovny.

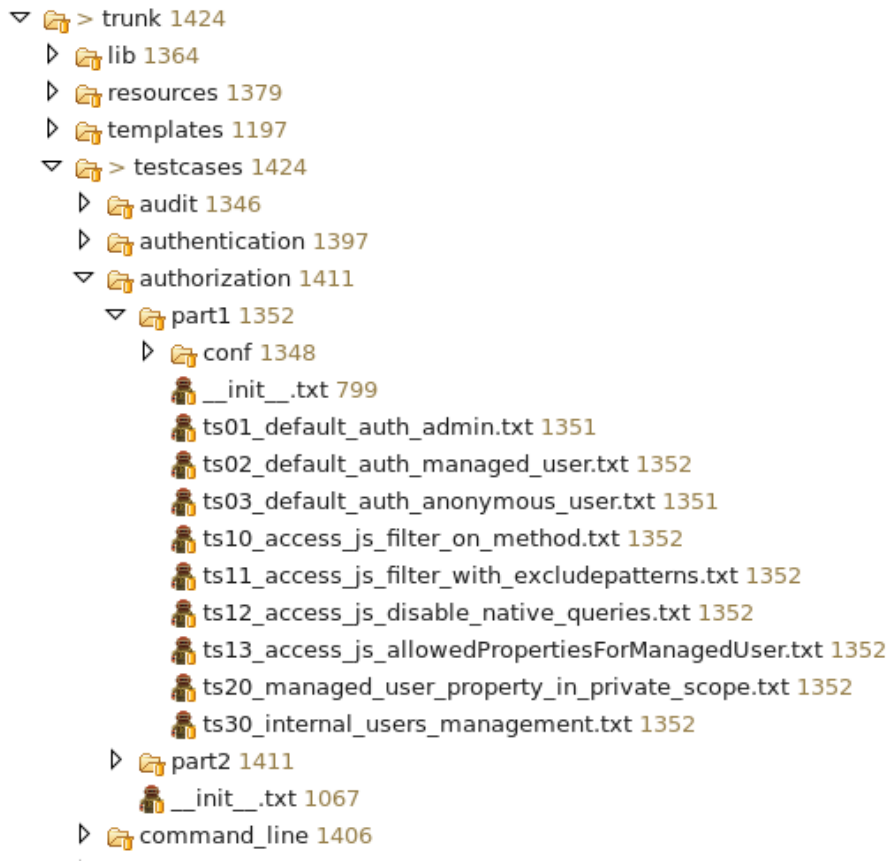
Pomocí Robot Frameworku a vlastních knihoven jsme byli schopni psát relativně rychle, jednoduše a jednoduše čitelné testy. Obrovská výhoda tohoto frameworku spočívá v jeho jednoduchosti - celý framework je tzv. Keyword driven, což znamená, že jednotlivé funkce jsou zabaleny do klíčových slov, které jsou pro člověka jednoduše srozumitelné. Příklad takového testu najdete na obrázku 2.

```
Description:
We test some hooks in router.json
We will just add some messages in the LOG.
To go through those hooks we will create a user and read it.

Documentation link:
http://openidm.forgerock.org/doc/integrators-guide/index.html#script-places


*** Settings ***
Resource        openidm_keywords.txt
Test Timeout    ${TIMEOUT}

*** Test Cases ***
ScriptFromRouter
    [Documentation]  scripts launched from router.json
    [Tags]  smoke
    # those actions should trigger every hooks defined in router.json
    create_jberg
    get_url     managed/user/jberg
    delete_jberg

    # hooks should have called scripts that wrote messages to the log
    grep_file_count_is_positive     ${FIRST_LOG_FILE}   I am in INFO log in the script router-log-onFailure
    grep_file_count_is_positive     ${FIRST_LOG_FILE}   I am in INFO log in the script router-log-onRequest
    grep_file_count_is_positive     ${FIRST_LOG_FILE}   I am in INFO log in the script router-log-onResponse

    [TearDown]  clean_openidm_users


```

Obrázek 2: Ukázka jednoduchého testu

Na tomto jednoduchém testu jde dobře vidět struktura těchto testů. V sekci Settings si importujeme knihovnu s klíčovými slovy, která v našem případě obsahuje všechna klíčová slova jak dodávaná s Robot Frameworkem, tak námi vytvořená a také si inicializujeme proměnnou Test Timeout. V sekci Test Cases již najdeme jednotlivé testy. V našem případě test nazvaný ScriptFromRouter.

Nyní se budeme zabývat tím, jak se jednotlivé testy dají škálovat a spouštět. Celý strom testů tvoří jednu velkou sestavu testů, tzv. Test Suite. Každý jeho podstrom tvoří také Test Suite. A každý další podstrom je také Test Suitem. Takto to pokračuje až na úroveň listů. Každý list (soubor s příponou txt) je sám o sobě Test Suite. Až uvnitř tohoto listu se nachází sekce Test Case, kde jsou definovány jednotlivé testy. Příklad takovéto struktury najdete na obrázku 3, kde každý adresář a každý soubor (vyjma __init__.txt - o tomto souboru bude ještě řeč.) představuje jeden TestSuite.



Obrázek 3: Strom Test Suite

Výše popsaná struktura nám tedy dovoluje spouštět jednotlivé Test Suite na libovolné úrovni původního Test Suite (na obrázku 3 adresář testcases). Pokud nám přesto takováto struktura nestačí, můžeme každému testu přiřadit klíčové slovo Tags - značku. Pomocí těchto značek si můžeme vytvářet jednotlivé skupiny testů. Na obrázku 2 vidíme, že test je označen značkou smoke.

Na následujícím obrázku (Obr. 4) vidíte příklady, jak lze spouštět jednotlivé Test Suite, respektive skupiny testů. První příkaz spustí Test Suite authorization, tedy i všechny Test Suite z part1 a part2 z obr. 3. Druhý příkaz spustí všechny Test Suite, které se nacházejí v testcases/authorization/part1/ a začínají na ts01 - tedy spustí se pouze jediný Test Suite a to ts01_default_auth_admin.txt (z obr. 3). Třetí příkaz spustí všechny testy v Test Suite testcases, které jsou označené značkou smoke. A čtvrtý, poslední příkaz spustí všechny testy v Test Suite testcases, kromě těch, které mají značku opendj.

```
pybot --argumentfile ./arguments.txt --variablefile ./variables.py --suite testcases.authorization .
pybot --argumentfile ./arguments.txt --variablefile ./variables.py --suite testcases.authorization.part1.ts01* .
pybot --argumentfile ./arguments.txt --variablefile ./variables.py --suite testcases --include smoke .
pybot --argumentfile ./arguments.txt --variablefile ./variables.py --suite testcases --exclude opendj .
```

Obrázek 4: Příklady spouštění testů

Nyní se ještě vraťme k inicializačním __init.txt__ souborům. Každý Test Suite může, ale nemusí, mít tento inicializační, ve kterém se dají definovat akce společné pro celý tento Test Suite. Konkrétně v našem případě jsme inicializačních souborů využívali pro nachystání testovacího prostředí. Toto nachystání prostředí zahrnuje instalaci OpenIDM a dalších aplikací potřebných pro tento Test Suite, příkladem může být aplikace OpenDJ [7], což je Open Source implementace LDAP serveru společnosti ForgeRock AS.

A proč právě Robot Framework? Pro Robot Framework jsme se rozhodli z několika důvodů:

1. Protože je jednoduše škálovatelný a testy jsou lépe organizované, jednodušeji se spravují a není nutné pokaždé spouštět všechny testy, protože někdy potřebujeme provést pouze kontrolu omezené funkčnosti, například v důsledku opravení nějaké chyby, která rozbíjela jenom jednu z funkcí produktu.

2. Dalším aspektem, který nahrával Robot Frameworku oproti jiným kandidátům, je to, že veškerá funkcionalita je skrytá za klíčovými slovy, a proto mohou jednotlivé testy vyvíjet i technicky méně zdatní lidé - člověk nemusí být programátor, aby zvládl napsat test.

3. Robot Framework sám o sobě obsahuje modul pro vytváření zpráv (reportů) o výsledcích testování a dokonce se dají do této generované zprávy přidat i výsledky manuálního testování, pokud nemáme všechny testy automatizované. Což je nesporně také velkou výhodou.

4.3 Reverzní inženýrství Contract Test Frameworku

Poté, co byla velká většina OpenIDM funkčních testů automatizována, jsem byl převelen na projekt reverzního inženýrství Contract Test Frameworku, který spadá pod OpenICF. Tento projekt byl převzat a oživen z dnes již neexistující společnosti Sun Microsystems, kde byl vyvíjen za účelem testování jednotlivých konektorů a naneštěstí nebyl plně dovyvinut a nijak zdokumentován, kromě poznámek v samotném zdrojovém kódu. Nicméně mi bylo řečeno, že framework je ve stavu, kdy je relativně použitelný.

Contract Test Framework je napsán v jazyce Java a pro spouštění jednotlivých testů využívá kombinaci testovacího frameworku TestNG a scriptovacího jazyku Groovy. TestNG se stará o řízení běhu testování, tedy o spouštění jednotlivých testovacích tříd a vyhodnocování testů a jazyk Groovy je zde využíván pro automatické generování hodnot atributů a pro poskytování různých konfiguračních objektů.

Nejprve jsem začal tak, že jsem tento framework propojil na stávající projekt LDAP konektoru, spustil si jednu instanci LDAP serveru - konkrétně OpenDJ od společnosti ForgeRock - a nakonfiguroval Contract Test framework tak, aby spustil své testy na tomto LDAP konektoru (neboli otestoval LDAP konektor). Výsledkem byla zpráva, která říkala, které testy byly spuštěny a jak dopadly - zda byl test úspěšný či nikoliv nebo jestli byl přeskočen. U neúspěšných testů byla přidána zpráva, která obsahovala také detailní informace, kde došlo k selhání. A právě tuto informaci jsem použil jako odrazový můstek pro další krokování zdrojového kódu. Našel jsem si, kde se daný test začíná vykonávat a postupně zjišťoval, co která metoda dělá. Když jsem měl zhruba představu, co daný test dělá, zdokumentoval jsem jeho průběh a také co (jakou funkcionalitu) se vlastně daný test snaží otestovat. Tímto jsem ve výsledku dosáhl toho, že jsem měl zdokumentovány všechny testy a věděl jsem, co který test dělá (testuje) a jak to dělá.

V průběhu krokování jsem si všiml, že spousta testů končí neúspěšně proto, že jim chybí konfigurační objekty a také proto, že některé hodnoty atributů jsou sice automaticky generovány, ale jejich hodnota

není správná. Například když hodnota atributu, který reprezentuje Common Name atribut, byla vygenerována jako řetězec šesti libovolných velkých písmen, přičemž validní hodnota vypadá například následovně: ”cn=Jan Novy,dc=example,dc=com”.

Vzhledem k tomu, že Contract Test Framework obsahoval i ukázky těchto konfiguračních objektů, začal jsem experimentovat a postupně přicházel na to, jak tyto objekty mají vypadat a jak zajistit, aby byla vygenerována správná hodnota atributu.

Výsledkem mého snažení tedy byla dokumentace, která obsahovala popis, co a jak daný test vlastně testuje a také jsem zdokumentoval, jak nakonfigurovat testy tak, aby skutečně testovaly funkcionalitu daného konektoru a nehlásily selhání testu, protože chybí nadefinovat některý z atributů. Tuto dokumentaci jsem následně předal vedoucímu vývojovému týmu, kterému tato dokumentace poslouží jako podklad pro opravu stávajících testů, vývoj nových a rozšíření funkčnosti LDAP konektoru.

Po dokončení toho úkolu bylo rozhodnuto, že mám podobným způsobem připravit konfigurační soubory pro další z konektorů a sice ScriptedSQL konektoru, který se používá pro připojení k různým implementacím SQL databází. Protože jsem již měl zkušenosti s konfigurací, myslel jsem si, že úkol bude jednodušší, ale ukázalo se, že abych dosáhl co největší úspěšnosti testů, musel jsem doimplementovat některé z funkcionalit konektoru.

Na závěr této části uvádím ukázku zdrojového kódu jednoho z testů a také ukázku jednoho konfiguračního objektu.

```
connector {
    host = '__configureme__'
    port = 389
    principal = '__configureme__'
    credentials = new GuardedString('__configureme__'.toCharArray())
    baseContexts = [ baseContext ] as String[]
    usePagedResultControl = true // We do not have a VLV index.
    uidAttribute = 'entryDN' // Sun DSEE 6.3 does not support entryUUID
}
```

Obrázek 5: Ukázka konfiguračního objektu

```java
/**
 * <p>
 * Non readable attributes should _not_ be returned by default
 * </p>
 * <p>
 * API operations for acquiring attributes: <code>GetApiOp</code>
 * </p>
 */
@Test(dataProvider = OBJECTCLASS_DATAPROVIDER)
public void testNonReadable(ObjectClass objectClass) {
    if (ConnectorHelper.operationsSupported(getConnectorFacade(),
            objectClass, getAPIOperations())) {
        Uid uid = null;
        try {
            ObjectClassInfo oci = getObjectClassInfo(objectClass);

            // create a new user
            Set<Attribute> attrs = ConnectorHelper.getCreateableAttributes(
                    getDataProvider(), oci, getTestName(), 0, true, false);
            // should throw UnsupportedObjectClass if not supported
            uid = getConnectorFacade().create(objectClass,
                    attrs, getOperationOptionsByOp(objectClass, CreateApiOp.class));

            // get the user to make sure it exists now
            ConnectorObject obj = getConnectorFacade().getObject(
                    objectClass, uid,
                    null/* GET returned by default attributes*/);

            assertNotNull(obj,"Unable to retrieve newly created object");

            // check: non readable attributes should not be returned by
            // default
            for (Attribute attr : obj.getAttributes()) {
                if (!ConnectorHelper.isReadable(oci, attr)) {
                    String msg = String
                            .format(
                                    "Non-readable attribute should not be returned by default: %s",
                                    attr.getName());
                    assertTrue( !ConnectorHelper.isReturnedByDefault(
                            oci, attr),msg);
                }
            }
        } finally {
            if (uid != null) {
                // delete the object
                getConnectorFacade().delete(objectClass, uid,
                        getOperationOptionsByOp(objectClass, DeleteApiOp.class));
            }
        }
    } else {
        printSkipTestMsg("testNonReadable", objectClass);
    }
}
```

Obrázek 6: Ukázka testu jednoho ze speciálních atributů konektoru

5 Znalosti a dovednosti získané s průběhu studia uplatněné v průběhu praxe

Při práci na zadaných úkolech jsem využil znalostí z předmětu Skriptovací programovací jazyky a jejich aplikace, kde jsem se naučil používat skriptovací jazyk Python. Těchto znalostí jsem široce využil při automatizaci testů OpenIDM. Dále jsem také využil znalostí jazyku Java nabytých v předmětu Programovací jazyky I. Tyto znalosti se hodily při reverzním inženýrství Contract Test Frameworku a jeho testů - tento framework je napsán v Javě. Protože OpenIDM i OpenICF jsou napsány v Javě a Java je multiplatformní, bylo potřeba oba produkty testovat na mnohých platformách a právě zde jsem využil znalostí Microsoft Windows Server z předmětu Správa Windows systémů a také znalostí základů Unix/Linux systémů z předmětu Operační systémy. Vzhledem k tomu, že jednou z primárních funkcí OpenIDM je provisioning do různých databázových systémů, využil jsem znalostí z předmětů Databázové a informační systémy a Teorie zpracování dat, protože bylo často potřeba provádět různé SQL dotazy, občas vytvořit nějaký SQL skript pro vytvoření či modifikaci testovací databáze.

6 Znalosti scházející studentovi v průběhu praxe

Hned z počátku jsem narazil na neznalost problematiky identity managementu. Následně jsem zjistil, že neznám operační systém Linux (Fedora, Ubuntu, CentOS, RHEL) tak, jak jej potřebuji znát pro výkon testování na této platformě. Dalším pro mě neznámým operačním systémem byl Solaris. Z počátku byl také trochu problém v neznalosti jazyka JavaScript a s tím spojený JSON (JavaScript Object Notation), který je v OpenIDM široce využíván, poté co jsem se naučil základy tohoto jazyka, problém se zdál být vyřešen. OpenIDM komunikuje pomocí REST interface a toto byla také jedna z věcí, se kterou jsem se setkal poprvé. Další z věcí, která byla trochu problém, byla neznalost týmové práce při vývoji většího systému a nástrojů pro správu kódu - SVN. Dalšími znalostmi, které jsem dosud postrádal, byly znalosti technik a procesů testování, konkrétně ve firmě používaný agilní přístup vývoje a testování.

7 Závěr

Tato část se bude zabývat dosaženými výsledky a celkovým hodnocením absolvované praxe.

7.1 OpenIDM a výsledky automatizace

Díky testování produktu OpenIDM jsem se seznámil s identity managementem, zjistil, k čemu se identity management používá a jaké problémy řeší. Protože je OpenIDM vyvíjeno jako Open Source projekt, byla to zároveň také dobrá zkušenost s tím, jak se Open Source produkt vyvíjí a jakou mají Open Source projekty pozici na trhu. Když už se bavíme o vývoji Open Source produktu, chtěl bych také zmínit, že toto byla moje první zkušenost s vývojem projektu větších rozměrů, kde se na tomto vývoji podílí různé týmy. Také jsem získal velmi cenné zkušenosti s tím, jak se takovéto projekty v praxi řídí.

Další oblastí, kde jsem získal hodně cenných zkušeností, je oblast testování software obecně. Zde jsem se naučil, do jaké fáze vývoje projektu takovéto testování zapadá, jaké jsou způsoby a druhy testování a v neposlední řadě také jak se toto testování řídí.

Nyní se přesuňme k automatizaci testování OpenIDM a jejímu zhodnocení. Na vývoji automatizace se dohromady podíleli 4 lidé. Za necelé 2 měsíce jsme byli schopni navrhnout automatizační framework a automatizovat přibližně 90% definovaných testů. V průběhu automatizace jsme také začali objevovat některé oblasti, které ještě testy nebyly pokryty, a pro tyto oblasti také navrhli testy a následně je zautomatizovali. V době psaní této práce existuje přibližně 235 automatizovaných testů, přičemž doba potřebná pro spuštění všech testů činí okolo 30 minut. Když jsme tyto testy prováděli manuálně, testování zabralo přibližně 20 man-day. Pokud bychom tuto hodnotu přepočetli na minuty a porovnali s dobou běhu automatizovaných testů, dostaneme rozdíl 9570 minut! Neboli jeden kompletní testovací cyklus je oproti manuálnímu testování 319 krát rychlejší!

Takováto automatizace nám umožnila spouštět testy pokaždé, když některý z vývojářů provede nějaké změny v kódu aplikace. Tímto jsme schopni velice rychle rozpoznat, jestli nově přidaný kód nějakým způsobem negativně ovlivnil chování aplikace (rozbil nějakou její část nebo má nový kód nečekané ”vedlejší účinky”).

7.2 Reverzní inženýrství Contract Test Frameworku

Vzhledem k tomu, že byl Contract Test framework převzat z jiné společnosti a já jsem byl jediný, kdo na tomto úkolu pracoval, byl tento úkol velmi náročný, protože jsem neměl žádnou oporu v nikom, kdo by tento framework již znal a byl by mi schopen vysvětlit fungováni tohoto frameworku. Také to bylo poprvé, kdy jsem se dostal k reverznímu inženýrství v takovémto rozsahu, který, jak je obecně známo, není jednoduchý. Nicméně úkol jsem přijal jako velkou výzvu a test mých schopností.

Výsledkem práce, jak již bylo popsáno v kapitole 4.3, byla dokumentace obsahující popis, jaké oblasti dané testy pokrývají, jak dobře jsou tyto oblasti pokryty a jakým způsobem se dané testy provádějí. Tato dokumentace byla následně předána vývojovému oddělení, které má za úkol dovyvinout tento framework a doimplementovat další testy.

7.3 Blog

Jedním z výsledků mé práce jsou také blogy a snaha podporovat komunitu okolo ForgeRock produktů. V době psaní této práce jsem již publikoval několik blogů, které popisují integraci OpenIDM s některými datovými úložišti. Těmito blogy se snažím podporovat komunitu, protože OpenIDM je mladý a rychle se rozvíjející produkt a mnoho věcí se mění a některé věci nejsou úplně zdokumentovány nebo je implementována nová funkcionalita, ale zatím není oficiálně podporovaná. Odkazy na blogy najdete na následující www adrese: http://blog.profiq.cz/author/lfolta/.

7.4 Celkové zhodnocení praxe

Tato praxe byla pro mne velkým přínosem. Díky této praxi jsem získal velmi cenné zkušenosti v oblastech řízení vývoje velkého projektu, práce ve větším týmu a spolupráce v rámci jednotlivých týmů. Tato praxe mě také naučila zodpovědnosti při týmové práci a dodržování dohodnutých konvencí při psaní kódu, protože kdyby si každý člen týmu psal kód dle vlastních konvencí, kód by byl velice nepřehledný a těžko by se udržoval. Další významnou zkušeností bylo reverzní inženýrství, kdy jsem musel pochopit celý kód poměrně složitého frameworku. Práce na reverzním inženýrství mě také naučila, že ne vždy je potřeba vyvíjet produkty od píky, ale stojí za zvážení, zda se nám vyplatí investovat čas do reverzního inženýrství. V případě Contract Test Frameworku si myslím, že jsme zvolili dobře a ten investovaný měsíc práce se opravdu vyplatil, protože nám přinesl vcelku fungující framework s velkým množstvím fungujících testů. V neposlední řadě jsem se díky absolvování této praxe dostal oboru testování software, který mě velice zaujal a naučil mne, že testování software je stejně důležité jako jeho vývoj.

Literatura

[1] profiq s.r.o. *Profiq* [online]. (c) 2010 [cit. 2013-04-21]. Dostupné z: http://www.profiq.cz

[2] FORGEROCK AS. *ForgeRock* [online]. © 2010 – 13 [cit. 2013-04-21]. Dostupné z: http://forgerock.com/

[3] FORGEROCK AS. *OpenIDM project* [online]. (c) 2011 – 2013 [cit. 2013-04-21]. Dostupné z: http://openidm.forgerock.org/

[4] FORGEROCK AS. *OpenICF project* [online]. (c) 2011 – 2013 [cit. 2013-04-21]. Dostupné z: http://openicf.forgerock.org/

[5] OpenIDM 2.1.0 Installation Guide. CRAIG, Mark, Lana FROST, Paul BRYAN, Andi EGLOFF, Lazslo HORDOS a Matthias THRISTL. FORGEROCK AS. *OpenIDM 2.1.0 Installation Guide* [online]. March 28, 2013 [cit. 2013-04-21]. Dostupné z: http://docs.forgerock.org/en/openidm/2.1.0/install-guide/index.html

[6] OpenIDM 2.1.0 Integrator's Guide. CRAIG, Mark, Lana FROST, Paul BRYAN, Andi EGLOFF, Lazslo HORDOS, Matthias THRISTL a Anders ASK?SEN. FORGEROCK AS. *OpenIDM 2.1.0 Installation Guide* [online]. March 28, 2013 [cit. 2013-04-21]. Dostupné z: http://docs.forgerock.org/en/openidm/2.1.0/integrators-guide/index.html

[7] FORGEROCK AS. *OpenDJ project* [online]. (c) 2011 – 2013 [cit. 2013-04-21]. Dostupné z: http://opendj.forgerock.org/

[8] *Robotframework* [online]. [cit. 2013-04-21]. Dostupné z: http://code.google.com/p/robotframework/

[9] *Python Programming Language* [online]. © 1990-2013 [cit. 2013-04-21]. Dostupné z: http://www.python.org/

[10] *OpenIDM* [online]. © 2010–13 [cit. 2013-04-21]. Dostupné z: http://forgerock.com/what-we-offer/open-identity-stack/openidm/

[11] *What is Identity* [online]. [cit. 2013-04-28]. Dostupné z: http://www.karingroup.com/eng/about/what_is_identity.pdf